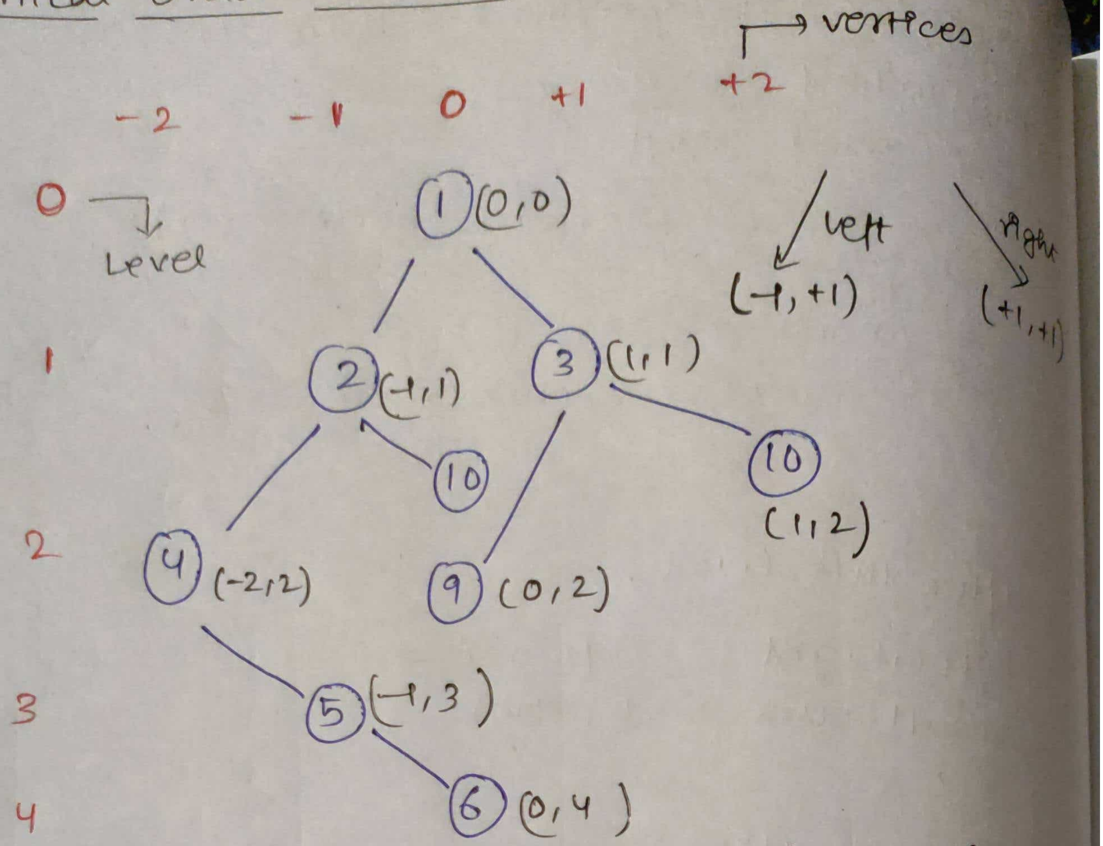
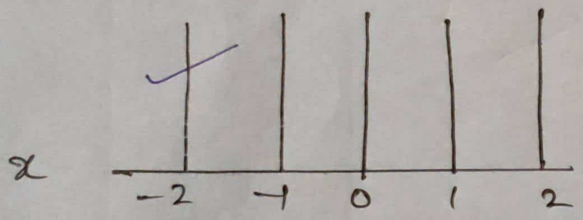
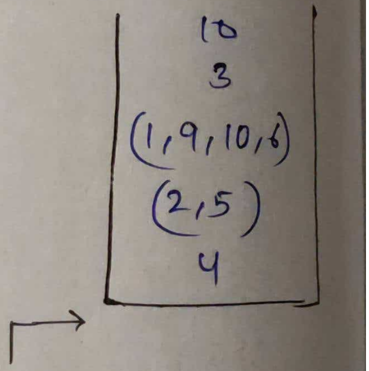
#Vertical Order Traversal :-



- We need to traverse the vertical level wise in ascending order.





visit (-2) → store 4 in data structure.

visit (-1) → " (2,5)

visit (0) → (1, 9, 10, 6) (sorted order)

" (+1) → 3.

" (+2) → 10

Queue

(node, v, lev)

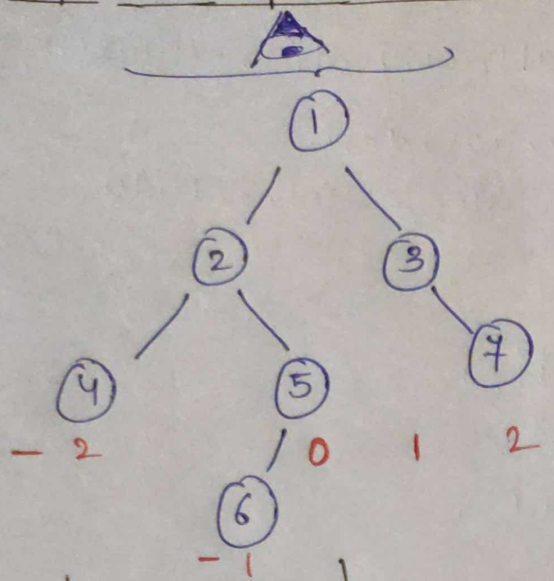
vertical ↑ levels ↑ multi-nodes

map<int, map<int, multiset<int>>>

Code:-

```
vector<vector<int>> verticalTraversal(TreeNode * root){
    map<int, multiset<int>>> nodes;
    queue<pair<TreeNode*, pair<int,int>>> todo;
                          ↳ store (node, vertical, lev)
      ↳ traversing nodes
    todo.push({root, {0,0}});
    while (!todo.empty()){
        auto p= todo.front();
        todo.pop();
        TreeNode *node = p.first;
    int x= p.second.first, y= p.second.second;
    nodes[x][y].insert(node->val);
      if (node->~~val~~left){
        todo.push({node->left, {x-1, y+1}});
      }
      if (node->right){
        todo.push({node->right, {x+1, y+1}});
      }
    }
    vector<vector<int>> ans;
      for (auto p: nodes){
            vector<int> col;
        for (auto q: p.second){
      col.insert(col.end(), q.second.begin(),
                          q.second.end());
        }
      ans.push_back(col);
    }
    return ans;
}
```

Top view of a Binary Tree:-



O/P: 4 2 1 3 7

We will be using the level order traversal technique to solve the problem.

→ line concept (Vertical Order Traversal)

2 → 7	5
-2 → 4	1
1 → 3	4
-1 → 2	2
0 → 1	3

map(line, node)

~~(6,-1)~~
~~(7,2)~~
~~(5,0)~~
~~(4,-2)~~
~~(3,+1)~~
~~(2,-1)~~
~~(1,0)~~

Queue

node : (level-wise traverse)	~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~7~~	~~6~~
	~~0~~	~~-1~~	~~1~~	~~-2~~	~~0~~	~~2~~	~~-1~~

Already visited

- For getting top-view from left to right, check line wise sequence, (from map)

-2 → 4
-1 → 2
0 → ~~3~~1
1 → 3
2 → 7

Expected Output

code:-

```
vector<int> topView (Node * root){
        vector<int> ans;
        if(root == NULL) return ans;
        map<int,int> mp; // storing line & node
      queue <pair< Node *, int>> q;
      q.push ({root,0});
     while(!q.empty()){
          auto it = q.front();
          q.pop();
         Node * node = it.first;
         int line = it.second;
  if(mp.find(line) == mp.end()){
        mp[line] = node->data;
     }
  if (node->left != NULL){
        q.push ({node->left, line-1});
      }
  if (node->right != NULL) {
       q.push ({node->right, line+1});

    }
}
  for (auto it: mp){
          ans.push_back (it.second);
          }
     return ans;
  }
```